Estimating the Class Prior in Positive and Unlabeled Data through Decision Tree Induction

Jessa Bekker and **Jesse Davis**
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract

For tasks such as medical diagnosis and knowledge base completion, a classifier may only have access to positive and unlabeled examples, where the unlabeled data consists of both positive and negative examples. One way that enables learning from this type of data is knowing the true class prior. In this paper, we propose a simple yet effective method for estimating the class prior, by estimating the probability that a positive example is selected to be labeled. Our key insight is that subdomains of the data give a lower bound on this probability. This lower bound gets closer to the real probability as the ratio of labeled examples increases. Finding such subsets can naturally be done via top-down decision tree induction. Experiments show that our method makes estimates which are equivalently accurate as those of the state of the art methods, and is an order of magnitude faster.

Introduction

Traditional approaches to supervised learning for binary classification assume a fully labeled training set consisting of positive and negative examples. In many applications, a learner may only have access to positive examples and unlabeled examples, which may be either positive or negative. This special case of semi-supervised learning is commonly called positive and unlabeled (PU) learning.

The following are four illustrative examples of domains where PU data may arise. First, medical records usually only list the diagnosed diseases for each person and not the diseases the person does not have. However, many diseases, such as diabetes, are often undiagnosed (Claesen et al. 2015b). Therefore, the absence of a diagnosis does not mean that the patient does not have a disease. Second, the task of knowledge base (KB) completion is also inherently a positive and unlabeled problem. Automatically constructed KBs are necessarily incomplete and only contain true facts. The truth values of the facts not included in the KB are unknown and not necessarily false (Galárraga et al. 2015; Neelakantan, Roth, and McCallum 2015). Third, similar to the above scenario, there are many specialized but incomplete gene databases (Elkan and Noto 2008). Finally, text classification can also be characterized by positive and unlabeled data (Lee and Liu 2003; Liu et al. 2003). For example, classifying a user's web page preferences could use bookmarked pages as positive examples and all other pages as unlabeled ones.

Several approaches exist to deal with positive and unlabeled data. The most straightforward one is to assume that all the unlabeled data are negative and simply apply standard machine learning techniques (Neelakantan, Roth, and McCallum 2015). A second approach is to select some of the unlabeled examples that are very different from the positively labeled ones and label them as negative. A classifier is then learned using the given positive examples and inferred negative examples (Liu et al. 2002; Li and Liu 2003; Yu, Han, and Chang 2004; Yu 2005; Li et al. 2009; Nguyen, Li, and Ng 2011). A third approach is to employ an evaluation metric that only uses positive, or positive and unlabeled data (Muggleton 1996; Lee and Liu 2003; Claesen et al. 2015a). Using this metric, one can tune for the best class weights or regularization settings (Lee and Liu 2003; Liu et al. 2005; Mordelet and Vert 2014; Claesen et al. 2015c). A final approach is to explicitly consider the class prior. It can be used to either adapt algorithms to incorporate this information during learning (Denis 1998; Liu et al. 2003; Zhang and Lee 2005; Denis, Gilleron, and Letouzey 2005; Elkan and Noto 2008) or as a preprocessing step to assign weights to the unlabeled examples (Elkan and Noto 2008). Because the class prior is often not known, several methods were proposed in the last decade to estimate it from the positive and unlabeled data (Elkan and Noto 2008; du Plessis and Sugiyama 2014; du Plessis, Niu, and Sugiyama 2015; Jain, White, and Radivojac 2016; Jain et al. 2016; Ramaswamy, Scott, and Tewari 2016)

This paper focusses on the last task and presents a novel method for estimating the class prior. It exploits the "selected completely at random" assumption which says that each positive example has a constant probability c to be selected to be labeled, c is referred to as the *label frequency*. With this assumption, the label frequency functions as a proxy for the class prior. Our method is based on two main insights. First, the label frequency holds in subdomains of the attributes and the probability for examples to be labeled in a subdomain provides a lower bound for it. Second, subdomains with a higher probability of labeled examples yield better estimates. This last insight suggests that finding discriminative partitions in the data will yield good estimates of

the label frequency. Finding such partitions is exactly what top-down decision tree induction algorithms do to select a test for an internal node. Therefore, we propose a decision tree-based approach for estimating the label frequency. With extensive experiments, we show that this simple technique gives stable estimates that are equivalently accurate to the state-of-art methods but is an order of magnitude faster. The implementation and extra materials are available online.[1]

Problem Setup

Positive and unlabeled (PU) learning attempts to learn a binary classifier while only having access to positively labeled and unlabeled data. An example is represented by $\{x, y, s\}$, where x are its attributes, y the true class and s the label. Only positive examples are labeled: $s = 1 \Rightarrow y = 1$, and unlabeled examples $s = 0$ can be of any class y. The number of positive, negative, positively labeled, and total number of examples in a data (sub)set are P, N, L and T respectively.

A common assumption in PU learning is the "selected completely at random" assumption, which states that the labeled dataset is selected completely at random from the entire positive set, i.e., the label and the attributes are conditionally independent given the true class (Denis 1998; De Comité et al. 1999; Liu et al. 2003; Zhang and Lee 2005; Elkan and Noto 2008). Hence, any positive example has exactly the same probability to be selected to be labeled as any other positive example, independent of its attributes:

$$\Pr(s = 1|x, y = 1) = \Pr(s = 1|y = 1). \tag{1}$$

We refer to this probability as the *label frequency* $c = \Pr(s = 1|y = 1)$. From the assumption follows the following property (Elkan and Noto 2008):

$$\Pr(y = 1|x) = \frac{1}{c} \Pr(s = 1|x). \tag{2}$$

Elkan and Noto (2008) showed that knowing the label frequency c greatly simplifies PU learning because it enables using Equation (2) in one of three ways. First, a probabilistic classifier can be trained to predict $\Pr(s = 1|x)$ and the output probabilities can be adjusted using Equation (2). Second, the same classifier and Equation (2) can be used to weight the unlabeled data, and then a different classifier can be trained on the weighted data. Third, Equation (2) can be used to modify a learning algorithm. For example, count-based algorithms, like tree induction and naive Bayes, only consider the number of positive and negative examples in attribute-conditioned subsets of the data. In PU data these numbers are expected to be $\hat{P} = L/c$ and $\hat{N} = T - \hat{P}$ (Denis 1998; De Comité et al. 1999; Denis et al. 2003).

The label frequency, or equivalently the class prior $\alpha = \Pr(s = 1)/c$ can be obtained in three ways: 1) from domain knowledge, 2) by estimating it from a small fully labeled data set (De Comité et al. 1999), or 3) by estimating it directly from the PU data (Elkan and Noto 2008; du Plessis, Niu, and Sugiyama 2015; Jain, White, and Radivojac 2016; Ramaswamy, Scott, and Tewari 2016). We present a simple, yet effective method for the last option.

[1] https://dtai.cs.kuleuven.be/software/tice

Bounding and Estimating the Label Frequency Using Tree Induction

In this work, we aim to estimate the label frequency by considering subdomains of the attributes based on partial assignments. For ease of derivation, we assume discrete variables. The use of subdomains is possible because of the "selected completely at random" which implies label frequencies being equal in any subdomain A. This can be derived using the law of total probability and Equation (1).

$$\begin{aligned}
&\Pr(s = 1|x \in A, y = 1) \\
&= \sum_{x'} \Pr(s = 1, x = x'|x \in A, y = 1) \\
&= \sum_{x' \in A} \Pr(s = 1, x = x'|x \in A, y = 1) \\
&= \sum_{x' \in A} \Pr(s = 1|x = x', y = 1) \Pr(x = x'|x \in A, y = 1) \\
&= \Pr(s = 1|y = 1) \sum_{x' \in A} \Pr(x = x'|x \in A, y = 1) \\
&= \Pr(s = 1|y = 1) = c.
\end{aligned} \tag{3}$$

Therefore, the label frequency is the ratio of the probabilities to be labeled and to be positive in any subdomain A:[2]

$$c = \frac{\Pr(s = 1|x \in A)}{\Pr(y = 1|x \in A)}. \tag{4}$$

If A is a positive subdomain $\Pr(y = 1|x \in A) = 1$, the probability of being labeled in this subdomain equals the label frequency. In general, the probability is a lower bound for the label frequency because probabilities are at most 1:

$$c \geq \Pr(s = 1|x \in A). \tag{5}$$

Label Frequency Lower Bound from Data Subset

Using (5), we can use any attribute-conditioned subset of the data, with L labeled and T total examples, to estimate a lower bound on c. Naively, this would be $c \geq L/T$. However, because of the stochastic nature of the labeling, more positive examples might be labeled than expected. Therefore, we include an error term $\epsilon = 1/2\sqrt{(1 - \delta/\delta T)}$ that shrinks with the sample size T. The error term is derived from the one-sided Chebyshev inequality. This inequality provides a lower bound on the probability of the number of labeled examples L exceeding the expected number μ by at least λ, based on the labeling variance σ^2. The probability lower bound is set to $\delta = \sigma^2/(\sigma^2 + \lambda^2)$.

$$\Pr\left(L \geq \mu + \lambda\right) \leq \frac{\sigma^2}{\sigma^2 + \lambda^2}$$

$$\Pr\left(L \geq \mu + \sqrt{\frac{(1 - \delta)\sigma^2}{\delta}}\right) \leq \delta.$$

Labeling positive examples follows the Binomial distribution, because, each positive example in the subdomain is independently labeled with a probability c. Therefore, $\mu = cP$

[2] Analogous to the derivation of (2) in (Elkan and Noto 2008)

and $\sigma^2 = c(1-c)P$. Substituting μ and σ, and using $P \leq T$, gives a probabilistic lower bound for c:

$$\Pr\left(L - cT \geq \sqrt{\frac{(1-\delta)c(1-c)T}{\delta}}\right) \leq \delta$$

$$\Pr\left(c \leq \frac{L}{T} - \sqrt{\frac{(1-\delta)c(1-c)}{\delta T}}\right) \leq \delta. \quad (6)$$

The error term depends on c. But by using probability properties: $c(1-c) < 0.25$, a lower bound with probability at least $1-\delta$ can be calculated using only T and L:

$$\Pr\left(c \leq \frac{L}{T} - \frac{1}{2}\sqrt{\frac{1-\delta}{\delta T}}\right) \leq \delta. \quad (7)$$

Interesting Subsets

Using the lower bound in Equation (7) to estimate the true label frequency c, requires calculating it in a subset where the bound is tight. This is the case in large (almost) purely positive subsets. The positive probability is directly proportional to the labeled probability, therefore, positive subdomains are expected to have a high ratio of labeled examples. Mostly positive subdomains are therefore likely to be found when looking for highly labeled regions in the data.

If a subset is selected because it has a high number of labeled examples, it is likely that the number of labeled examples will exceed the expected number cP. Therefore, the bound in Equation (7) does not hold in such a subset. This issue is resolved by using independent datasets to 1) identify interesting subdomains and 2) calculate the lower bound. To intuitively see that it is resolved, consider datasets $D1$ and $D2$ which are independently sampled from the probability distribution $\Pr(x, y)$. $D1$ is labeled following the "selected completely at random" assumption by throwing a die for each positive example. Now we look for a subdomain that has a high proportion of labeled examples in $D1$. Next, $D2$ is labeled using the same procedure, therefore each positive example in the previously found subdomain has the same probability to be labeled as any other example. The probabilities are unaffected by the subdomain search procedure. Labeling $D2$ first does not change the search procedure, ensuring unaffected probabilities for $D2$ that are usable for calculating the lower bounds.

Tree Induction for Label Frequency Estimation

We propose a novel method TI*c*E (*Tree Induction for c Estimation*) for estimating the label frequency from PU data, which is summarized in Alg. 1. The algorithm is based on the insights of the previous sections: It splits the dataset into two separate sets, looks for interesting, i.e., likely positive, subdomains using one set and estimates c using the other set by taking the tightest lower bound that is calculated in the interesting subdomains. Looking for pure subsets in the data is also the objective of decision tree induction, therefore TI*c*E looks for pure labeled subsets by inducing a decision tree, considering the unlabeled data as negative.

Folds The separate datasets are referred to as the *tree data* and the *estimation data*, the former is used for inducing the tree and the later for estimating the label frequency using the subdomains. To make robust estimates, the process is repeated for k folds: the training data is divided into k random equal-sized subsets and in every fold, 1 subset is used as tree data, the rest as estimation data. The estimate is the average of the estimates of the folds.

Max c_{low} The label frequency is estimated as the maximum lower bound calculated in multiple subsets of the estimation data. The maximum invalidates the lower bound of Equation (7) because although each lower bound has a probability $1-\delta$ of holding, the probability of all the lower bounds holding is smaller. The maximum could be avoided by only calculating the lower bound on one subdomain. The most promising subdomain is selected solely based on the *tree data*. This would ensure a correct lower bound with probability $1-\delta$. However, we argue that the maximum would work better in practice. The lower bound is very loose and is only likely to get close to the true c in the rare case of a completely positive subset. By only taking one lower bound, the chance that a lower bound is calculated in a purely positive subset decreases. Especially with a low label frequency, it is hard to predict which subdomains are positive. Moreover, the goal of TI*c*E is not to find a lower bound, it is to find a close estimate which does not need to be a lower bound. This decision is evaluated in the experiments section.

Split The objective of standard decision tree induction is finding pure nodes. Here, only pure *positive* nodes are of interest. To reflect this, the *maximum biased estimate for the proportion of positives (max-bepp)* score is used (Blockeel, Page, and Srinivasan 2005). It selects the split that gives the subset with the highest bepp: $\frac{P}{T+k}$, where the parameter k acts like the Laplace parameter to prefer larger subsets.

Tighter bound with $\hat{c}$ Equation (7) assumes the worst case of $c = 0.5$, making the error term larger than necessary in most cases. An initial estimate c_{prior}, used in Equation (6) makes a more accurate estimate. Therefore, TI*c*E first induces a tree and to estimate c_{prior} and then repeats the process to estimate $\hat{c}$, using c_{prior}.

Choosing δ To calculate lower bounds, the parameter δ needs to be supplied. Its optimal value depends on the application and the data dimensions. δ can be chosen by supplying the minimum number of examples T_R that should be required to calculate a lower bound with some error term ϵ: $\delta = \frac{1}{1+4\epsilon^2 T_R}$.

We propose a simple rule for choosing δ, which we evaluate in the experiments section. Big datasets, that contain more than 10,000 examples require 1,000 examples to update c_{low} with an error of $\epsilon = 0.1$. Smaller datasets need one tenth of their data: $T_R = \min[1000, 0.1T]$. Therefore:

$$\delta = \max\left[0.025\ ,\ \frac{1}{1+0.004T}\right]. \quad (8)$$

Algorithm 1: TIcE (k, M, f)

Input: k: max-bepp parameter
M: maximum number of splits
f: tree/estimation folds
Result: $\hat{c}$

1 $\hat{c} \leftarrow 0.5$;
2 **for** $i = 0$; $i < 2$; i++ **do**
3 $\quad \hat{c}s \leftarrow []$;
4 $\quad$ **for** *(tree data, estimation data)* $\in f$ **do**
5 $\quad\quad \delta \leftarrow \max\left[0.025,\ \frac{1}{1+0.004T(\text{estimation data})}\right]$;
6 $\quad\quad c_{\text{best}} \leftarrow \frac{L(\text{estimation data})}{T(\text{estimation data})}$;
7 $\quad\quad q \leftarrow [(\text{tree data, estimation data})]$;
8 $\quad\quad$ **for** $j = 0$; $j < M$ **and** $|q| > 0$; j++ **do**
9 $\quad\quad\quad (S_t, S_e) \leftarrow \underset{(S_t,S_e)\in q}{\text{argmax}} \left[\frac{L(S_t)}{T(S_t)} - \sqrt{\frac{\hat{c}(1-\hat{c})(1-\delta)}{\delta T(S_t)}}\right]$;
10 $\quad\quad\quad q.\text{remove}\left((S_t, S_e)\right)$;
11 $\quad\quad\quad a^* = \underset{a\in\text{atts}(S_e)}{\text{argmax}} \underset{v\in\text{Dom}(a)}{\max} \frac{L(\{S_t : a=v\})}{T(\{S_t : a=v\})+k}$;
12 $\quad\quad\quad$ **for** $v \in Dom(a^*)$ **do**
13 $\quad\quad\quad\quad q.\text{append}\left((\{S_t : a^* = v\}, \{S_e : a^* = v\})\right)$;
14 $\quad\quad\quad\quad c_{\text{low}} = \frac{L(\{S_e:a^*=v\})}{T(\{S_e:a^*=v\})} - \sqrt{\frac{\hat{c}(1-\hat{c})(1-\delta)}{\delta T(\{S_e:a^*=v\})}}$;
15 $\quad\quad\quad\quad c_{\text{best}} \leftarrow \max\left(c_{\text{best}}\ ,\ c_{\text{low}}\right)$;
16 $\quad\quad \hat{c}s.\text{append}(c_{\text{best}})$;
17 $\quad \hat{c} \leftarrow \text{avg}(\hat{c}s)$;

Speed Top-down decision tree induction is an efficient algorithm but can be further sped up by limiting the number of splits to a constant. Limiting the splits with optimal quality preservation is achieved by executing the splits in a best-first order. For this, the nodes need to be scored to indicate which one is most likely to result in good subsets. TIcE uses the lower bound provided by the node data, which needs to be calculated in the tree data to prevent overfitting.

Complexity The worst case time complexity of TIcE is $\mathcal{O}(mn)$, with n the number of examples n and m the number of attributes, assuming that each attribute's domain size is at most d. The size of the queue q can never exceed $(d-1)\cdot M$, therefore, all queue operations (lines 9, 10 and 13) have a constant worst-case time complexity. Nodes are recursively split in lines 9 to 15. Finding the best attribute to split on (line 11), requires going over all available attributes and all the tree data in the node, which has a complexity $\mathcal{O}(mn/|f|) = \mathcal{O}(mn)$. The estimation data will be split on the found attribute (needed in lines 13 and 14), which has complexity $\mathcal{O}(n(|f|-1)/|f|) = \mathcal{O}(n)$. Lines 12 to 15 have complexity $\mathcal{O}(d)$ because for each value of the domain, constant-time operations are executed. The total complexity of splitting a node is thus $\mathcal{O}(nm + n + d) = \mathcal{O}(nm)$. The loops of lines 2,4 and 8 each have a constant number of iterations: 2, $|f|$ and M and, thus, do not alter the complexity.

Related Work

The first work to consider the label frequency explicitly is Elkan and Noto (2008). Their insight is that a probabilistic classifier trained on PU data is expected to classify positive examples as positive with a probability c. To estimate c, they train a classifier on part of the data and predict the probabilities of the positively labeled examples in the other part. $\hat{c}$ is the average of the predicted probabilities. Another estimator they proposed was the highest predicted probability by the trained classifier on an example in the validation set. They discarded this estimator because of its high variability. This estimator is very related to TIcE, but two important differences make TIcE more reliable. First, Elkan and Noto's estimator only uses one data point for estimation. Second, the parameters of the model, and thus the prediction, are based on the training data, while TIcE computes its estimate from the validation data.

Recently several mixture proportion estimation methods have been developed to estimate the class prior in PU data. Their intuition assumes the case-control scenario, where the positive dataset is seen as a distribution and the unlabeled data as the mixture of the positive distribution and an unknown negative distribution (Blanchard, Lee, and Scott 2010; Sanderson and Scott 2014; Scott 2015). The next paragraphs describe such methods in more detail.

du Plessis and Sugiyama (2014) aim to find the class prior using partial matching. They estimate $\hat{\alpha}$ by minimizing the Pearson divergence between $\Pr(\mathbf{x})$ and $\alpha \Pr(\mathbf{x}|\mathbf{y}=\mathbf{1})$, where the two distributions are estimated from the unlabeled and positively labeled data respectively. If the support of the two classes is not disjoint, this method always overestimates the class prior. To correct this, du Plessis, Niu, and Sugiyama (2015) propose to use a biased penalty that heavily penalizes estimates $\hat{\alpha}$ that unrealistically imply that in some subdomains of the distributions $\alpha \Pr(\mathbf{x}|\mathbf{y}=\mathbf{1}) > \Pr(\mathbf{x})$.

Jain et al. (2016) propose to model the mixture of the positive and negative distributions using kernels and model the positive distribution by reweighting the kernels. The weights represent how much the positive distribution contributes to each kernel, therefore, the class prior is the sum of the weights. They find the optimal weights by maximizing the class prior simultaneously with the likelihood of the mixture and positive distribution in the unlabeled and positively labeled data. Ramaswamy, Scott, and Tewari (2016) propose a similar method but instead of using the likelihood, they use the distance between kernel embeddings. These methods assume the labeled data to have no false positives. Jain, White, and Radivojac (2016) allow for noisy labeled data by also considering the labeled data to be generated by a mixture.

Experiments

We aim to gain insight in the performance of TIcE. First, we check if in practice it is better to take the maximum of lower bounds or to use one lower bound. Second, we evaluate our method for setting δ. Finally, we compare TIcE to other class prior estimation algorithms.

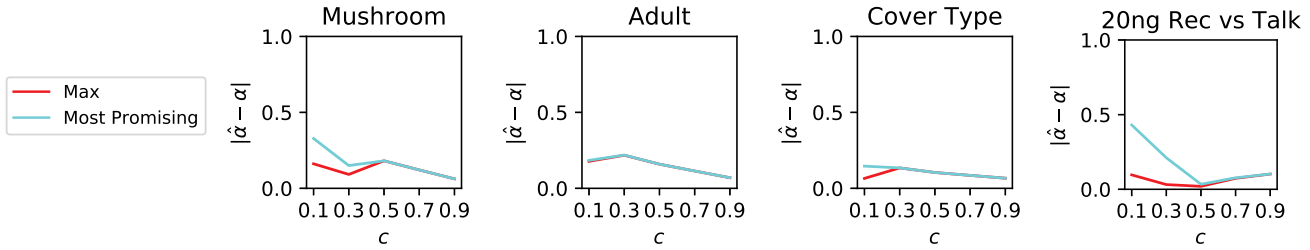

Figure 1: **Taking the maximum lower bound vs most promising subdomain**, for representative datasets in the PU setting. The true label frequency is varied on the x-axis. The lower the error, the better. Using the maximum gives better results, especially for lower frequencies. All other results are available in the online appendix.

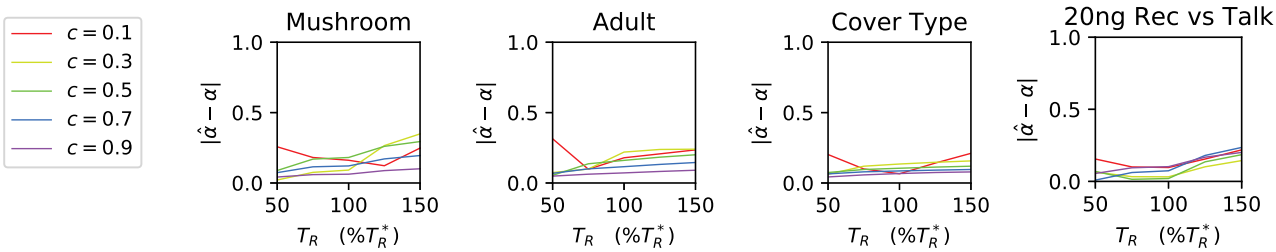

Figure 2: **Sensitivity to** T_R, for representative datasets in the PU setting. The number of examples to update c_{low} with an error $\epsilon = 0.1$ is varied on the x-axis. TIcE is not very sensitive to T_R. All other results are available in the online appendix.

Table 1: Datasets

Dataset	# Examples	# Vars	$\Pr(y = 1)$
Breast Cancer	683	9	0.350
Mushroom	8,124	21	0.482
Adult	48,842	14	0.761
IJCNN	141,691	22	0.096
Cover Type	536,301	54	0.495
20ng comp vs rec	5,287	200	0.450
20ng comp vs sci	5,279	200	0.450
20ng comp vs talk	4,856	200	0.401
20ng rec vs sci	4,752	200	0.499
20ng rec vs talk	4,329	200	0.450
20ng sci vs talk	4,321	200	0.451

Table 2: **Sensitivity to** T_R

%	$\lvert\hat{\alpha} - \alpha\rvert$ (all)	$\lvert\hat{\alpha} - \alpha\rvert$ (3 largest datasets)
50	0.092	0.084
75	0.089	0.079
100	0.090	0.095
125	0.135	0.108
150	0.170	0.127

Data

We use 11 real-world datasets that are summarized in Table 1. IJCNN was used for the IJCNN 2001 neural network competition[3] (Prokhorov 2001). All the others are UCI[4] datasets. Features were generated for the twenty newsgroups data (20ng) using bag of words with the 200 most frequent words, disregarding nltk stopwords. Binary classification tasks were defined by classifying pairs of general categories (computer, recreation, science, and talk). All the multivalued features were binarized and the numerical features were scaled between 0 and 1.

Methods

To create PU datasets from the completely labeled datasets, the positive examples are selected to be labeled with label frequencies $c \in [0.1, 0.3, 0.5, 0.7, 0.9]$. In addition to PU datasets, NU datasets were generated by labeling negative examples with the same frequencies. Per label frequency and class label, 5 different random labelings were executed. In total there are $11 \cdot 2 \cdot 5 \cdot 5 = 550$ settings.

We fixed all the hyperparameters of our method TIcE, because, in the considered context, no supervised validation dataset is available for tuning. The max-bepp parameter is $k = 5$ as in the original paper. The maximum number of splits is $M = 500$, which is expected to be large enough in any setting because of the best-first ordering. δ was set using Equation (8) on the estimation data in each fold. The number of folds is 5. All experiments are repeated 5 times with different random folds. If a node is split on a numerical feature, the range is split into 4 equal parts. When only the most promising subdomain is considered, this subdomain is selected by calculating lower bounds on the tree data. The implementation and additional results are available online.[5]

[3] Available on: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[4] `http://archive.ics.uci.edu/ml/`

[5] `https://dtai.cs.kuleuven.be/software/tice`

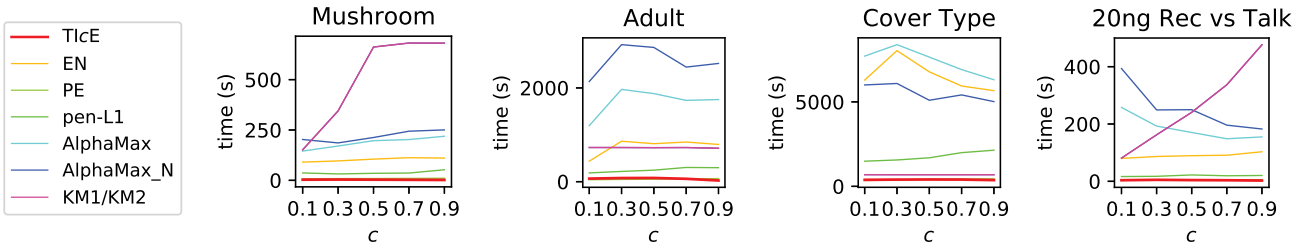

Figure 3: **Time comparison** between different methods, for representative datasets in the PU setting. The true label frequency is varied on the x-axis. TIcE is consistently fast. All other results are available in the online appendix.

Table 3: **Time comparison** between different methods.

Method	Average time per example (ms)
TIcE	**0.76**
PE	1.10
pen-L1	4.84
EN	21.96
AlphaMax	38.47
KM1/KM2	47.28
AlphaMax_N	52.92

We compared to the following class prior estimation methods that also make the "selected completely at random" assumption: EN (Elkan and Noto 2008), PE (du Plessis and Sugiyama 2014), pen-L1 (du Plessis, Niu, and Sugiyama 2015), KM1 and KM2 (Ramaswamy, Scott, and Tewari 2016), AlphaMax (Jain et al. 2016) and AlphaMax_N (Jain, White, and Radivojac 2016).[6] KM1 and KM2 cannot handle large datasets. Therefore, like the authors of those papers, we subsampled the datasets to have at most 2000 examples and repeated the process five times. The implementation also does not estimate a class prior on the smallest dataset Breast Cancer.

For all the experiments, the absolute error on the class prior $|\hat{\alpha} - \alpha|$ and CPU time were measured. The label frequency c is converted to the class prior with $\hat{\alpha} = L/(\hat{c}T)$. To evaluate using Equation (8) to set δ, its sensitivity to T_R is analyzed. To this end, in addition to using the proposed value $T_R^* = \min[1000, 0.1T]$, the experiments were also executed with $T_R \in \{0.5, 0.75, 1.25, 1.5\} \cdot T_R^*$.

All experiments are executed on a Red Hat Enterprise Linux ComputeNode with access to 24G RAM and 1 core of a Xeon E5-2680v2 CPU.

[6]The code for PE was taken from `http://www.mcduplessis.com/index.php/class-prior-estimation-from-positive-and-unlabeled-data/` and for KM1 and KM2 from `http://web.eecs.umich.edu/~cscott/code/kernel_MPE.zip`. The code for pen-L1 was the same as in (du Plessis, Niu, and Sugiyama 2015). The code for AlphaMax, AlphaMax_N and EN were the same as in (Jain, White, and Radivojac 2016). These were acquired through personal communication.

Results

Using the maximum of lower bounds often gives better estimates for the label frequency and is never worse than only using the lower bound of the "most promising" subdomain (Fig. 1). This is in line with our expectations. When little data is available, like for Breast Cancer, predicting the "most promising" subdomain is even more challenging. For Adult, both methods work fine because it has a high class prior.

Changing T_R does not change the estimates much, especially when the the dataset is large (Fig. 2 and Table 2). In fact, our rule (8) is usually overly conservative. We did not change the rule because that would be tuning on the test data.

When comparing the absolute error $|\hat{\alpha} - \alpha|$ between different methods, we see that TIcE is equivalently accurate as the state of the art (Fig. 4 and Table 4). Its average rank according to $|\hat{\alpha} - \alpha|$ is the second best, after KM2, and it has the lowest average absolute error. Note that it is also very stable, its predictions are never off much by much. This is reflected in its standard deviation, which is the lowest of all the methods. For comparison, look at the results of KM2. This is usually the most accurate estimator, but when it does not have access to many labels (e.g. 20ng Rec vs Sci, $c = 0.1$), it is biased too much towards 0.5 which results in large errors. Tice's stability is owed to two reasons: First, overestimates are rare because of the conservative lower bounds. Second, a tight lower bound is likely to be found because it is likely that some subdomain exists that is (close to) purely positive.

TIcE is a fast method, the other estimators that are equivalent or slightly worse (KM2, AlphaMax_N, AlphaMax) are more than an order of magnitude slower (Fig. 3 and Table 3). Speedwise, PE comes closest to TIcE, however, it gives less accurate estimates for the class prior. Note that KM1 and KM2 seem to have reasonable times for big datasets such as Cover Type, this is because it uses a subsample of the data instead of the full dataset.

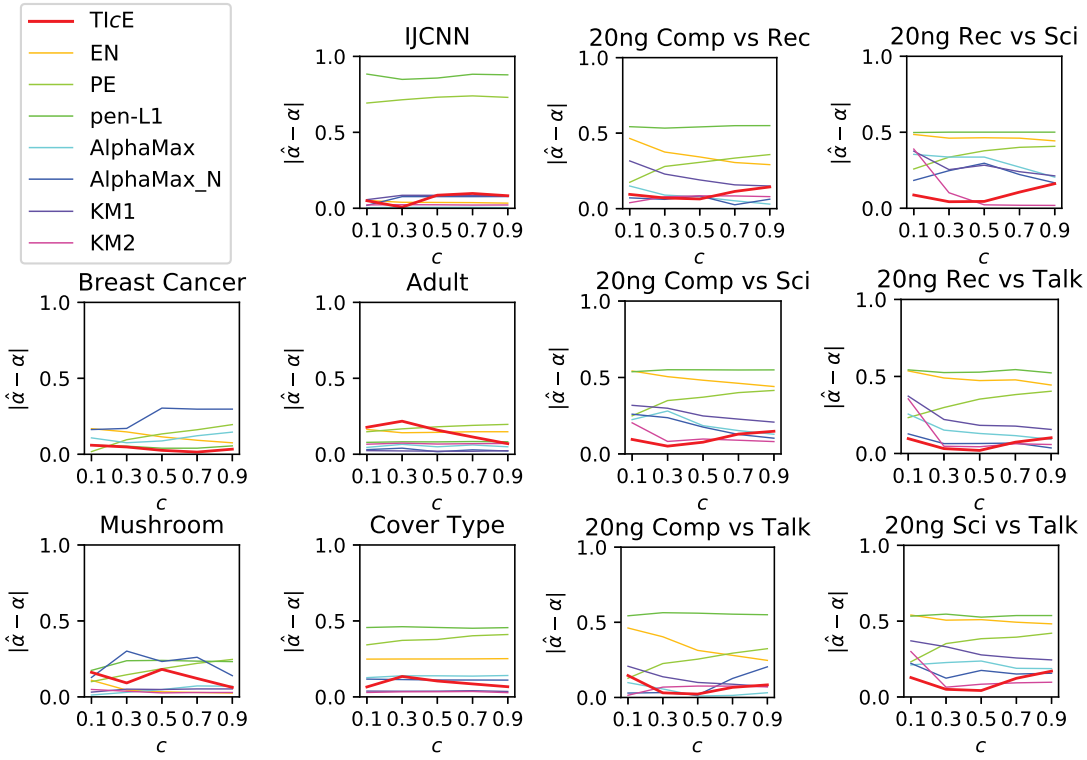

Figure 4: **Absolute class prior error comparison** between different methods, for PU datasets. The true label frequency is varied on the x-axis. The lower the error, the better. TIcE gives stable estimates, with an average error 0.09 and standard deviation 0.06. The plots for the NU datasets are available in the online appendix.

Table 4: **Absolute class prior error comparison** between different methods. TIcE is the second-best method when considering the rank in all settings. However, it has the best average performance and lowest standard deviation, therefore, it is more reliable.

Method	Average $\lvert\hat{\alpha} - \alpha\rvert$ rank +/- SD	Average $\lvert\hat{\alpha} - \alpha\rvert$ +/- SD
KM2	**2.83 +/- 2.03**	0.10 +/- 0.13
TIcE	3.22 +/- 1.82	**0.09 +/- 0.06**
AlphaMax_N	3.42 +/- 1.89	0.13 +/- 0.10
AlphaMax	3.51 +/- 1.38	0.12 +/- 0.09
KM1	4.04 +/- 1.19	0.11 +/- 0.09
EN	5.84 +/- 1.62	0.27 +/- 0.16
PE	6.16 +/- 1.39	0.29 +/- 0.14
pen-L1	6.88 +/- 1.86	0.37 +/- 0.20

Conclusions

In this paper, we propose a simple yet effective method for estimating the class prior. The method is based on the insight that the label frequency (which serves as a proxy for the class prior) is expected to be the same in any subdomain of the attributes. As a result, subsets of the data naturally imply lower bounds on the label frequency. The lower bounds will be tight when the subset belongs to a positive subdomain. Finding likely positive subdomains can easily be done using decision tree induction based on the PU data. Despite the simplicity of the method, it gives good and stable estimates. The experiments show that this method is equivalently accurate to the state of the art but an order of magnitude faster.

Acknowledgements

We thank Hendrik Blockeel and Adrian Dunne for their valuable advice. JB is supported by IWT (SB/141744). JD is partially supported by the KU Leuven Research Fund (C14/17/070, C22/15/015, C32/17/036) and FWO-Vlaanderen (G.0356.12, SBO-150033).

References

Blanchard, G.; Lee, G.; and Scott, C. 2010. Semi-supervised novelty detection. *JMLR* 2973–3009.

Blockeel, H.; Page, D.; and Srinivasan, A. 2005. Multi-instance tree learning. In *ICML*, 57–64.

Claesen, M.; Davis, J.; De Smet, F.; and De Moor, B. 2015a. Assessing binary classifiers using only positive and unlabeled data. *arXiv preprint arXiv:1504.06837*.

Claesen, M.; De Smet, F.; Gillard, P.; Mathieu, C.; and De Moor, B. 2015b. Building classifiers to predict the start of glucose-lowering pharmacotherapy using belgian health expenditure data. *arXiv preprint arXiv:1504.07389*.

Claesen, M.; De Smet, F.; Suykens, J.; and De Moor, B. 2015c. A robust ensemble approach to learn from positive and unlabeled data using svm base models. *Neurocomputing* 73–84.

De Comité, F.; Denis, F.; Gilleron, R.; and Letouzey, F. 1999. Positive and unlabeled examples help learning. In *ALT*, 219–230.

Denis, F.; Laurent, A.; Gilleron, R.; and Tommasi, M. 2003. Text classification and co-training from positive and unlabeled examples. In *ICML 2003 workshop: the continuum from labeled to unlabeled data*, 80–87.

Denis, F.; Gilleron, R.; and Letouzey, F. 2005. Learning from positive and unlabeled examples. *TCS* 70–83.

Denis, F. 1998. Pac learning from positive statistical queries. In *ALT*, 112–126.

du Plessis, M., and Sugiyama, M. 2014. Class prior estimation from positive and unlabeled data. *IEICE Transactions* 1358–1362.

du Plessis, M.; Niu, G.; and Sugiyama, M. 2015. Class-prior estimation for learning from positive and unlabeled data. *ML* 1–30.

Elkan, C., and Noto, K. 2008. Learning classifiers from only positive and unlabeled data. In *SIGKDD*, 213–220.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. 2015. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal* 707–730.

Jain, S.; White, M.; Trosset, M. W.; and Radivojac, P. 2016. Nonparametric semi-supervised learning of class proportions. *arXiv preprint arXiv:1601.01944*.

Jain, S.; White, M.; and Radivojac, P. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *NIPS*.

Lee, W. S., and Liu, B. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, 448–455.

Li, X.-L., and Liu, B. 2003. Learning to classify texts using positive and unlabeled data. In *IJCAI*, 587–592.

Li, X.-L.; Yu, P.; Liu, B.; and Ng, S.-K. 2009. Positive unlabeled learning for data stream classification. In *SDM*, 259–270.

Liu, B.; Lee, W.; Yu, P.; and Li, X.-L. 2002. Partially supervised classification of text documents. In *ICML*, volume 2, 387–394. Citeseer.

Liu, B.; Dai, Y.; Li, X.-L.; Lee, W.; and Yu, P. 2003. Building text classifiers using positive and unlabeled examples. In *ICDM*, 179–186.

Liu, Z.; Shi, W.; Li, D.; and Qin, Q. 2005. Partially supervised classification–based on weighted unlabeled samples support vector machine. In *ADMA*, 118–129.

Mordelet, F., and Vert, J.-P. 2014. A bagging svm to learn from positive and unlabeled examples. *Pattern Recognition Letters* 201–209.

Muggleton, S. 1996. Learning from positive data. In *ILP*, 358–376.

Neelakantan, A.; Roth, B.; and McCallum, A. 2015. Compositional vector space models for knowledge base completion. *ACL*.

Nguyen, M. N.; Li, X.-L.; and Ng, S.-K. 2011. Positive unlabeled leaning for time series classification. In *IJCAI*, 1421–1426.

Prokhorov, D. 2001. IJCNN 2001 neural network competition. *Slide presentation in IJCNN*.

Ramaswamy, H.; Scott, C.; and Tewari, A. 2016. Mixture proportion estimation via kernel embedding of distributions. In *ICML*.

Sanderson, T., and Scott, C. 2014. Class proportion estimation with application to multiclass anomaly rejection. In *AISTATS*.

Scott, C. 2015. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *AISTATS*.

Yu, H.; Han, J.; and Chang, K. C.-C. 2004. Pebl: Web page classification without negative examples. *TKE* 70–81.

Yu, H. 2005. Single-class classification with mapping convergence. *Machine Learning* 49–69.

Zhang, D., and Lee, W. 2005. A simple probabilistic approach to learning from positive and unlabeled examples. In *UKCI*, 83–87.